Photon Map Tricks

Per Christensen

Square USA / Pixar

SIGGRAPH Course #38, 2001

Contents

- Six “tricks” to make the photon map method faster, better, and easier to use
- Some tricks give up to 10x improvement

Overview

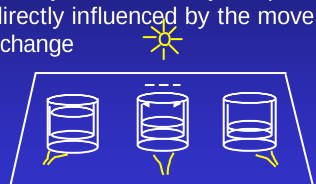
- 1 “trick” for less flicker in animations
- 1 “trick” for improved ease of use
- 3 “tricks” for improved speed
- 1 “trick” for improved accuracy

Trick #1: Less flickering animations

Frame-coherent random numbers

Goal

- When an object moves, only the photon paths directly influenced by the move should change



Problem

- If *one* sequence of random numbers is used for reflection/refraction/absorption probabilities, then:
- If *one* photon path length changes, *all* the following photon paths will change, too!

Solution

- Several solutions; the simplest+best:
- Use a separate random number sequence for each photon. Seed can depend on just photon number
- Reduces flickering to ~ 1/10

Origins

- Nuclear physics: Monte Carlo simulation of neutron scattering (1950s)
- References: [Goertzel58, Spanier69]

Trick #2: Improved ease of use

Automatically computed maximum search radius for lookups

Background: radiance estimates

Two ways to determine photon density:

- Fixed number of photons (determine area)
- Fixed area (find all photons within area)

$E = \Sigma P_n / A$

Background: radiance estimates (2)

In practice, we use a combination of the two:

- Fixed number of photons in high and medium density regions
- Fixed area in sparse regions

Combination:

search for max *n* photons within max radius *r*

Effects of the max search radius

- If the max radius is set too high: large parts of the kd-tree will be searched in sparse regions, only to result in a very dim radiance. Waste of time!
- If the max radius is set too small: the area either contains a photon or not. "Polka dot" pattern!

Optimal value: tedious trial-and-error

Computing the max search radius

- Switch between number-limited and area-limited when the density is low enough that inaccuracies are not important:

  $r = 1/\pi \sqrt{n P_{max} / L_t}$

  – n = number of photons
  – P_{max} = max photon power
  – L_t = threshold radiance (e.g. 5% of white)

Speedup

- Using an optimal max search radius is up to 10 times faster than using no max search radius (in sparse regions)
- This trick makes it trivial to find that optimal max radius

Trick #3: Improved speed

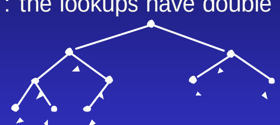
Iterative lookups instead of recursive

Photon map lookup

- Find *n* photons closest to point *p* (within max radius *r*)
- Usually done with a recursive traversal of the kd-tree
- Normally trivial to rewrite recursive algorithm as iterative; faster

Photon map lookup (2)

- BUT: the lookups have double recursion



- Question: is it nevertheless possible to rewrite as an iterative algorithm?

Photon map lookup (3)

- Yes! (using two small auxiliary arrays)
- Algorithm is in course notes
- Speedup: up to 25%

Trick #4: Improved speed

Final gathering using precomputed irradiance estimates

[Christensen00]: "Faster photon map global illumination", Journal of Graphics Tools, 4(3):1-10

Observations

1. Final gathering is the bottleneck – especially the photon map lookups
2. Rays from different final gathers hit nearly the same place: almost identical photon map lookups are repeated again and again

Faster final gathering

- Precompute irradiance at all (or some) photon positions; store with photons
- During final gathering, use precomputed irradiance at the nearest photon with similar normal (Surf. are Voronoi diag.)

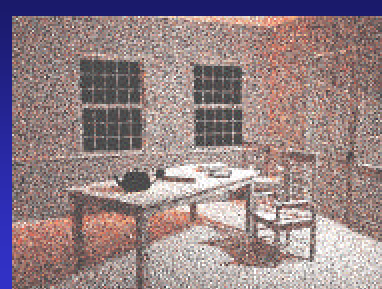
Example

- >1M polygons
- 1024 x 768 pixels, up to 16 samples/pixel
- Linux PC with 733MHz Pentium, 540MB

Example: classic ray tracing

Classic ray tracing: 2.5 min. (soft shadows, textures)

Example: photon map



500,000 photons. Tracing: 8 sec, sorting: 4 sec

Example: irradiance w/o precomp.

Irradiance estimates at image sample points (200 photons)

Example: precomputed irradiance

Precomputed irradiance estimates at 125,000 photon positions.
Precomputation: 21 sec.

Example: radiance

Radiance estimates (this is what the final gather 'sees')

Example: complete image

Complete image. Final gathering: 22 min -> 3 min.
Total render time: 25 min -> 6 min.

Speedup

- Makes final gathering 5-8 times faster by removing the bottleneck
- Cost: precomputation time (<2%); extra memory (28%)

Other applications

- Participating media: ray marching
- Importance ("importons" [Peter98])

Variations

- Compute irradiances "on demand" (instead of precomputation)
- Compute more than ¼ irradiances where there is high gradient in photon density
- Store radiance instead of irradiance:
  - saves many texture lookups
  - less accurate

More variations

- Other quantities can be stored with photons to save time
- See course notes

Trick #5: Improved accuracy

Combining irradiance estimates from several photon maps

Why have several photon maps?

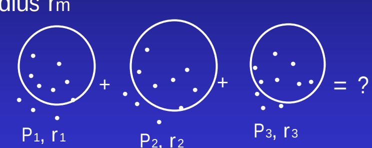
- Usually not necessary
  - photon map fits within memory of one machine
  - Photon map can be shared among processors (shared memory)
- But in extreme cases several photon maps can be necessary (computed in parallel)

Assumption

- Photons are evenly distributed among the photon maps

Combining lookup results

- Each lookup produces a power P_m and a radius r_m



- How to combine them?

Combining lookup results (2)

- First idea: $E_{total} = \Sigma\,(P_m / \pi\, r_m^2)$
- Less statistical variation: combine powers and radii separately:

  $E_{total} = (\Sigma\, P_m) / \pi\, \mathrm{ave}(r_m)^2$
- Why better? Non-linear

Trick #6: Improved speed

Photon tracing using importance

Goals

1. Reduce time to trace photons in very complex scenes
2. Reduce space, i.e # stored photons
3. No visible bias
4. No mixing of photons with high and low power in the photon map – polka dots

Importance strategy

- First: trace importance particles from the camera position [Peter98]
- During photon tracing: shoot few photons to unimportant regions

Previous work using importance

- [Peter98]: reduced # traced and stored photons, but mixed low- and high-energy photons
- [Suykens00, Keller00]: reduced # stored photons, but not # traced photons

Proposal: adaptive photon emission

- Divide directions from light source into small strata
- In each stratum: first send few “feeler” photons
  - if no feeler photon reach anything important: just store them in photon map
  - If some feeler photons reach something important: emit many more photons within that stratum

Preliminary results

- Only 2 scenes
- Speedup: factor ~ 15
- No visible bias

To be done

- Experimental comparison with [Peter98]
  - bias
  - noise
- Extension to participating media

Conclusion

Six optimizations of the photon map method:

- Frame-coherent random numbers: reduce flicker to ~ 1/10
- Optimal r_{max}: speedup ≤ 10
- Iterative lookup: speedup ≤ 25%

Conclusion (2)

- Precomputing irradiance: speedup 5-8
- Improved combination of lookup results: higher accuracy (how much?)
- Photon tracing using importance: speedup ≤ 15

Future work

- Many more tricks to improve speed + accuracy await discovery ...
- Let's find them !

Acknowledgments

Many thanks to:

- Toshi Kato, Hitoshi Nishimura, Tadashi Endo, Tamotsu Maruyama, Jun Saito
- Everyone else at Square USA !